



# NOX: A C++ Nonlinear Solver Package

<http://software.sandia.gov/nox>

## The Abstract Layer

NOX is designed to be independent of the vector and matrix representation, the type of linear solver, and the interface to the function and Jacobian evaluations.

### NOX::Abstract::Vector Interface

- Initialization
  - $x = y$
  - $x = |y|$
  - $x_i = 1/y_i$  for  $i = 1$  to  $n$
  - $x_i = \gamma$  for  $i = 1$  to  $n$
- Clone (create a copy)
  - $y = x$
- Scaling
  - $x = \alpha x$
  - $x_i = x_i y_i$  for  $i = 1$  to  $n$
- Update
  - $x = \alpha a + \gamma x$ ,
  - $x = \alpha a + \beta b + \gamma x$
- Norm
  - $\|x\|_1, \|x\|_2, \|x\|_\infty$
  - $\|x\|_w$  (weighted norm)
- Dot
  - $x \cdot y$

All operations on the vector are via the proxy abstract vector.

### NOX::Abstract::Group Interface

- Iterate  $x$ 
  - Set  $x = y$
  - Update  $x = x + \alpha d$
  - Return  $x$
- Function value  $F = F(x)$ 
  - Compute
  - Return  $F$
- Clone
  - Create a copy

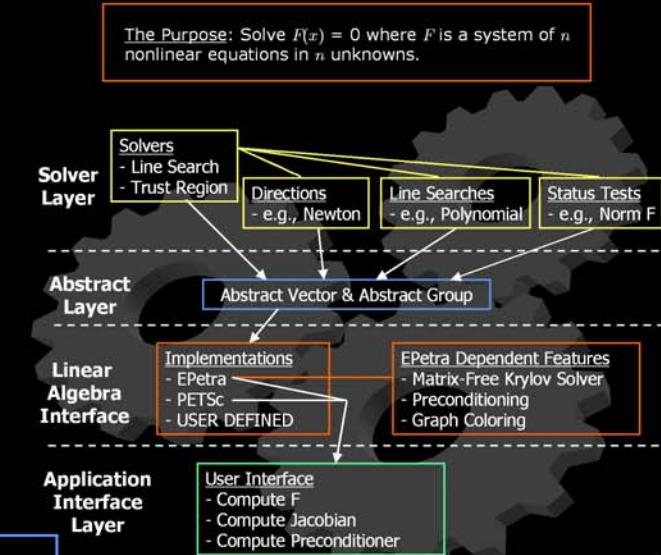
Access to the Function, the Jacobian, and the linear solver are all combined in the group object.

Gray denotes that support of these functions is optional.

#### NOX Developers

- Brett Bader (UC Boulder / Sandia)
- Russ Hooper (Sandia)
- Tammy Kolda (Sandia)
- Roger Pawlowski (Sandia)
- Eric Phipps (Sandia)
- Andy Salinger (Sandia)
- Joseph P. Simonis (WPI)

This work has been supported by the Department of Energy's ASCI program.



#### SAMPLE CODE

```
// Blah::Group derives from NOX::Abstract::Group.
// We assume that the interface to the problem is already set up,
// and that the x-value in this group is the initial guess
Blah::Group grp;

//(Relative norm must be less than 10^-4
NOX::StatusTest::NormF maxIter(grp, 1.0e-4);
// Max iterations must be less than 20
NOX::StatusTest::MaxIterations(20);
// This is a combination test that says at least one test must be satisfied
NOX::StatusTest::Combo combo(NOX::StatusTest::Combo::OR, normf, maxiters);

// Create the list of solver parameters
NOX::Parameter::List params;

// Set the solver
params.setParameter("Nonlinear Solver", "Line Search Based");

// Set the search direction
params sublist("Direction").setParameter("Method", "Newton");
params sublist("Direction").setParameter("Forcing Term Method", "Type 2");

// Set the line search
params sublist("Line Search").setParameter("Method", "Polynomial");
params sublist("Line Search").setParameter("Max Iters", 10);
params sublist("Line Search").setParameter("Interpolation Type", "Cubic");

// Create the solver
NOX::Solver::Manager solver(grp, combo, params);

// Solve the nonlinear system
NOX::StatusTest::StatusType status = solver.solve();

// Get the answer
grp = solver.getSolutionGroup();
```

We can combine and reuse different objects to create different solvers. In particular, we can easily swap in different line searches – including user-defined line searches.

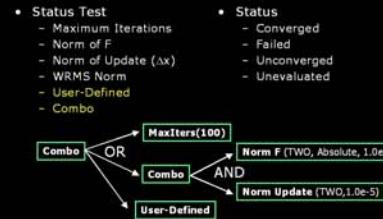
## Mix-n-Match Nonlinear Solver

- Solver
  - Line Search Based
  - Trust Region Based
  - Tensor Based
- Direction
  - Newton
  - Broyden
  - Steepest Descent
  - Tensor
  - User-Defined
- Line Search / B damping
  - Full Step
  - Backtrack
  - Polynomial/Quadratic
  - More-Thuente
  - Curvilinear
  - User-Defined



Any arbitrary combination of status tests can be used to control the termination of the nonlinear solver – including user-defined tests.

## Tinker Toy Status Tests

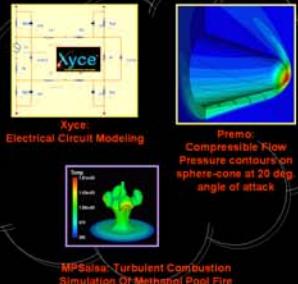


NOX is freely available via the GNU L-GPL license.

The project leads may be contacted by email at [tkolda@sandia.gov](mailto:tkolda@sandia.gov) and [rwpawlo@sandia.gov](mailto:rwpawlo@sandia.gov).

Though space limits us from listing the many internal and external collaborators we've been fortunate to work with, we'd still like to extend our sincere thanks to all of them for their valuable input which has greatly benefited this project.

## In-NOX-ulated Applications



## Useful Tools for Software Development



CVS - Repository for version tracking



Bonsai - Web GUI for CVS queries



Bugzilla - Bug tracking



Mailman - Archived mailing list